EXHIBIT 9

DOCKET NO: 0100157-00246

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

PATENT: 7,945,544

INVENTOR: DAVID A. FARBER
AND RONALD D. LACHMAN

FILED: OCT. 31, 2007 ISSUED: MAY 17, 2011

TITLE: SIMILARITY-BASED
ACCESS CONTROL OF DATA IN A
DATA PROCESSING SYSTEM

Mail Stop PATENT BOARD
Patent Trial and Appeal Board
U.S. Patent & Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450

PETITION FOR *INTER PARTES REVIEW* OF U.S. PATENT NO. 7,945,544 UNDER 35 U.S.C. § 312 AND 37 C.F.R. § 42.104

TABLE OF CONTENTS

TABLE OF AUTHORITIES

Page(s)

STATUTES

OTHER AUTHORITIES

I. MANDATORY NOTICES

A. Real Parties-in-Interest

EMC Corporation ("Petitioner") is the real party-in-interest.

B. Related Matters

The '544 patent is one of an extensive patent family of continuation and divisional applications. Exhibit 1008 shows the patent family, with patents in red and blue including the '544 patent being asserted in the litigation *PersonalWeb Technologies LLC v. EMC Corporation and VMware, Inc.* (No. 6:11-cv-00660-LED) (E.D. Tex.), served on December 16, 2012.

Petitioner is also seeking Inter Partes Review of related U.S. Patents Nos. 5,978,791, 6,415,280, 7,945,539, 7,949,662, and 8,001,096, and requests that they be assigned to the same Board for administrative efficiency. Moreover, there are several continuing applications related to this family that remain pending (shown on Exhibit 1008 in green). Because they share a common disclosure with the '544 patent, these applications may be used as a basis to present patentably indistinct claims that may issue prior to the determination of the PTAB in this or related Inter Partes Reviews. The issuance of indistinct claims is at least inconsistent with Rule 37 C.F.R. 42.73(d)(3)(i) and would be an "end-around" the reasonable number of substitute claims that are permitted in an IPR proceeding. Petitioner respectfully requests that the PTAB suspend from further prosecution, *sua sponte*, the

applications in this related family, including the applications shown on Exhibit 1008 in green and any further applications that may be filed that depend from this family of patents. If the PTAB determines that that suspension should be requested by written motion, permission to file such a motion is requested at this time.

C. Counsel

Lead Counsel: Peter M. Dichiara (Registration No. 38,005)

Backup Counsel: David L. Cavanaugh (Registration No. 36,476)

Petitioners will request authorization to file a motion for Cynthia Vreeland to appear *pro hac vice*. Ms. Vreeland has more than 20 years litigation experience, and has worked with Petitioner EMC on IP litigation matters for more than 10 years. As such, Ms. Vreeland has experience and established familiarity with the technology at issue in the case. Petitioners intend to file a motion seeking admission of Ms. Vreeland to appear *pro hac vice* when authorized to do so.

D. Service Information

Email: Peter Dichiara, peter.dichiara@wilmerhale.com

Post and Hand Delivery: WilmerHale, 60 State St., Boston MA 02109

Telephone: 617-526-6466 Facsimile: 617-526-5000

E. Certification of Grounds for Standing

Petitioner certifies pursuant to Rule 42.104(a) that the patent for which review is sought is available for *inter partes* review and that Petitioner is not barred or estopped from requesting an *inter partes* review challenging the patent claims on the grounds identified in this Petition.

II. OVERVIEW OF CHALLENGE AND RELIEF REQUESTED

A. Prior Art Patents and Printed Publications

Pursuant to Rules 42.22(a)(1) and 42.104 (b)(1)-(2), Petitioner challenges claim 1 of U.S. Patent No. 7,945,544 ("the '544 patent", Ex. 1001) as anticipated by or unpatentable in view of the following patents and printed publications:

1. Kantor, "The Frederick W. Kantor Contents-Signature System Version 1.22," FWKCS122.REF (August 10, 1993) ("Kantor", Ex. 1004).[1]

[1] Kantor's FWKCS user manual has been publicly and freely available continuously since August 1993. Kantor distributed the user manual with the FWKCS program as shareware and posted it online to electronic Bulletin Board Systems including "The Invention Factory" and "Channel 1" for an extended period of time, where it could be downloaded by anyone. As such, the document was accessible to others in the relevant community of BBS users and system operators. (*See* Kantor at 3; *see also* 158-59; Ex. 1004.)

2. S. Browne et al., "Location-Independent Naming for Virtual Distributed Software Repositories," University of Tennessee Technical Report CS-95-278 (Feb. 1995) ("Browne ", Ex. 1002).[2]
3. Albert Langer, "Re: dl/describe (File descriptions)," post to the "alt.sources" newsgroup on August 7, 1991 ("Langer", Ex. 1003)[3]

---

[2]The Browne February 1995 publication qualifies as prior art under 35 U.S.C. § 102(a), and is used in this petition because it includes illustrations which facilitate explanation of the grounds of the invalidity. Petitioner also has attached as exhibits and included in its claim charts two earlier versions of this publication – S. Browne et al., "Location-Independent Naming for Virtual Distributed Software Repositories," http://www.netlib.org/utk/papers/lifn/main.html (Nov. 11, 1994) (Exhibit 1006); and K. Moore et al., "An Architecture for Bulk File Distribution," Network Working Group Internet Draft (July 27, 1994) (Exhibit 1007). As Dr. Clark confirms in his declaration, the relevant disclosures are substantially the same. If the Patent Owner attempts to claim an earlier priority date of the challenged claims, Petitioner may rely on the earlier publications for invalidity, alone or in combination with the other references cited in this petition.

[3] Langer was made available on the "alt.sources.d" and "comp.archives.admin" newsgroup distribution lists on August 7, 1991. Both newsgroups were widely disseminated and readily accessible to the relevant technical community. Specifically, the "alt.sources.d" newsgroup was devoted to technical discussions

4. Woodhill et al., U.S. Patent No. 5,649,196, entitled "System and Method For Distributed Storage Management on Networked Computer Systems Using Binary Object Identifiers," filed Nov. 9, 1995 as a continuation of application 85,596, filed July 1, 1993 ("Woodhill", Ex. 1005).

B. There is a Reasonable Likelihood that at least One Claim of the '544 Patent is Unpatentable Under 35 U.S.C. §§ 102, 103

Section VI below explains how the above-cited patents and printed publications create a reasonable likelihood that Petitioner will prevail with the challenged claim. *See* 35 U.S.C. § 314(a). Indeed, that section together with the attached claim charts of Exhibits 1029, 1030, 1036, 1040 and the Declaration of Dr. Douglas Clark, a Professor of Computer Science at Princeton University ("Clark Decl."; Ex. 1009), demonstrate that the challenged claim is anticipated by, or unpatentable in view of, each of these references.

---

relating to the "alt.sources" source code repository. The "comp.archives.admin" newsgroup hosted discussions relating to computer archive administration. Therefore, an interested person would have been able to readily locate Langer among postings related to those subjects, both of which are in the same technical field as the '544 patent.

C. Relief Requested

Petitioner requests cancellation of claim 1, the challenged claim, as unpatentable under 35 U.S.C. §§ 102 and 103.

III. Claim Construction

The claim terms should be given their "broadest reasonable construction in light of the specification." 37 C.F.R. § 42.100(b).

The '544 patent includes the following constructions:

Claim Term	Construction
"data" and "data item"	"as used herein refer to sequences of bits. Thus a data item may be the contents of a file, a portion of a file, a page in memory, an object in an object-oriented program, a digital message, a digital scanned image, a part of a video or audio signal, or any other entity which can be represented by a sequence of bits" ('544 patent, col. 2, ll. 17-22, *see also* col. 2, ll. 28-31 (indicating "data items" can include files, directories, records in the database, objects in an object-oriented programming, locations in memory or on a physical device or the like"); Ex. 1001; '539 patent, claim 9 (indicating "sequences of bits" can include, in addition to the items articulated above, "a software product [or] a portion of a software product".)
"True Name, data identity, and data identifier"	"refer to the substantially unique data identifier for a particular item" ('544 patent, col. 6, ll. 20-22, see also col. 13, ll. 31-65 (describing mechanism for calculating True Name using MD

Claim Term	Construction
	hash function); Ex. 1001.)

IV. OVERVIEW OF THE '544 PATENT

A. Brief Description

The '544 patent is directed to data storage systems that use "substantially unique data identifiers" – based on all the data in a data item and only the data in the data item – to identify and access data items. (*See, e.g.*, '544 patent, Title, Abstract, and col. 1, ll. 44-49; Ex. 1001.) The patent uses these identifiers to perform basic file management functions, such as requesting and obtaining computer files or other data items, and eliminating unwanted duplicate copies of data items—admittedly old problems. (*See, e.g.*, '544 patent, Background of the Invention, col. 3, ll. 5-16; Ex. 1001.)

According to the patent, prior art systems identified data items based on their location or address within the data processing system. (*Id*. at col. 1, ll. 54-61.) For example, files were often identified by their context or "pathname," that is, information specifying a path through the computer directories to the particular file (*e.g.*, C:\My Documents\Law School\1L\TortsOutline.txt). (*Id*. at col. 1, ll. 66 – col.2, ll. 6.) The patent contends that all prior art systems operated in this manner: "In ***all*** of the prior data processing systems, the names or identifiers provided to

identify data items. . . are ***always*** defined relative to a specific context," and "there is ***no*** direct relationship between the data names and the data item." (*Id.* at col. 2. ll. 27-32, col. 2, ll. 40-41 (emphasis added).)

According to the patent, this prior art practice of identifying a data item by its context or pathname resulted in certain shortcomings. For example, with pathname identification, the same data name may refer to different data items, or conversely, two different data names may refer to the same data item. (*Id*. at col. 2, ll. 40-44.) Moreover, because there is no correlation between the contents of a data item and its pathname, there is no *a priori* way to confirm that the data item is in fact the one named by the pathname. (*Id*. at col. 2, ll. 45-48.) Furthermore, context or pathname identification may more easily result in the creation of unwanted duplicate data items, e.g., multiple copies of a file on a file server.[4] (*Id*. at col. 3, ll. 5-16.)

The '544 patent purports to address these shortcomings. (*Id*. at col. 3, ll. 31-45.) It suggests that "it is therefore desirable to have a mechanism . . . to determine a common and substantially unique identifier for a data item, using only the data in

[4] For example, Alice and Bob both download the same copy of the James Bond movie *Goldfinger*. Alice saves her copy at "C:\Movies\Bond\Goldfinger.mov", and Bob saves his copy at "C:\Videos\007\Bond-Goldfinger.mov".

the data item and not relying on any sort of context." ('544 patent, col. 3., ll. 32-36; Ex. 1001.) Moreover, "[i]t is further desirable to have a mechanism for reducing multiple copies of data items… and to have a mechanism which enable the identification of identical data items so as to reduce multiple copies." (*Id*. at col. 3, ll. 37-40.)

To do so, the '544 patent provides data identifiers that "depend[] on all of the data in the data item and only on the data in the data item." ('544 patent, col. 1, ll. 44-49, col. 3, ll. 51-56; Ex. 1001.) The preferred embodiments use either of the well-known MD5 or SHA message digest hash functions[5] to calculate a substantially unique identifier from the contents of the data item. (*Id.* at col. 12, l. 24 – col. 14, l. 2; Ex. 1001.) The system first computes the 16-byte (128-bit) message digest of the data item and then appends the size of the data item to

---

[5] A message digest or hash function transforms of a piece of data into a much shorter form by performing mathematical operations on its content. (*See, e.g.*, D. Banisar et al., The Third CSPR Cryptography and Privacy Conference at 509 (1993) (describing a message digest function as "a 128-bit cryptographically strong one-way hash function of the message" that is "somewhat analogous to a 'checksum' or CRC error checking code, in that it compactly 'represents' the message."); Ex. 1010.) The '544 patent admits that message digest functions were known. ('544 patent, col. 12, ll. 40-44; Ex. 1001.)

produce a 160-bit identifier. (*Id*. at Fig. 10A and col. 13, ll. 36-42.) The patent calls these context- or location-independent, content-based identifiers a "True Name" – a phrase admittedly "coined by the inventors." (U.S. Patent No. 6,415,280 Prosecution History, Response (Aug. 22, 2001), at 22; Ex. 1019.)

If a data item is large, it may include multiple components or "segments," representing the larger "compound data item." ('544 patent, col. 13, ll. 43-65; Ex. 1001.) A True Name identifier can be computed for each of the segments based on a hash of the contents of the segment. ('544 patent, col. 13, ll. 49–53; Ex. 1001.) Together, these segment identifiers form an "indirect block." (*Id*. at col. 13, ll. 53-56.) A True Name identifier is then computed for the compound data item as a whole based on a hash of the contents of the indirect block (*i.e*., a hash of the segment hashes). (*Id*. at col. 13, ll. 56-61.)

With these identifiers, the patent asserts, "data items can be accessed by reference to their identities (True Names) independent of their present location." (*Id*. at col. 33, ll. 28-30; *see also id*. at col. 33, ll. 49-51.) The actual data item corresponding to these location-independent identifiers may reside anywhere, e.g., locally, remotely, offline. (*Id*. at col. 33, ll. 30-38.) "Thus, the identity of a data item is independent of its name, origin, location, address, or other information not derivable directly from the data, and depends only on the data itself." (*Id.* at col. 3, ll. 56-59.)

In the preferred embodiments, the substantially unique identifiers are used to "augment" standard file management functions of an operating system.[6] (*Id*. at col. 6, ll. 25-32.) For example, a local directory extensions (LDE) table[7] is indexed by a pathname or contextual name of a file and also includes True Names for most files. (*Id.* at col. 8, ll. 28-36.) A True File registry (TFR) lists True Names, and stores "location, dependency, and migration information about True Files." (*Id*. at col. 8, ll. 37-39, 42-44.) True Files are identified in the True File registry by their True Names, and can be looked up in the registry by their True Names. (*Id*. at col. 8, ll. 40-42; col. 23, ll. 19-22.) This look-up provides, for each True Name, a list of the locations, such as file servers, where the corresponding file is stored. (*Id*. at col. 33, ll. 36-38; *see also id*. at col. 15, ll. 42-44.)

When opening or reading a file, the "Read File" mechanism "is aware of compound files and indirect blocks, and it uses [other] mechanisms to make sure that component segments are locally available. . . . When [a compound file] is

---

[6] The operating system is referred to as "existing" but is not named or otherwise identified.

[7] According to the patent, a LDE table is a data structure which provides information about files and directories in the system and includes information in addition to that provided by the native file system. ('544 patent, col. 8, ll. 28-36; Ex. 1001.)

opened only its indirect block is copied. When the corresponding file is read, the required component segments are realized and therefore copied." (*Id*. at col. 33, ll. 18-27.)

B. The Prosecution History of the '544 Patent

The '544 patent is based on an application originally filed on April 11, 1995. The initial claims were directed to determining segment identifiers for data items representing an audio signal, video signal or document content, and using the identifiers to determine if two data items are the same. For example, initial claim 1 of the application read as follows:

> 1. A method comprising:
>
> for a first data item comprising data representing an audio signal, video signal or document content, said first data item comprising a plurality of segments, determining a segment identifier for each of said segments, each said segment identifier being determined as a function of the data comprising the corresponding segment; and
>
> determining whether said first data item is the same as a second data item based at least in part on said segment identifiers.

(Application as Filed, Oct. 31, 2007, at 72; Ex. 1024**)** All claims were rejected as anticipated by Squibb (U.S. Pat. No. 6,816,872) or obvious in view of Squibb and Dyson (U.S. Pat. No. 5,050,212). (Office Action, July 2, 2010, at 3 and 5, Ex. 1025.)

In response, the applicants amended the claims at length. For example, initial claim 1 was amended as follows:

> 1. (Currently amended) A computer-implemented method, the method comprising:
>
> for a first data item comprising ~~data representing an audio signal, video signal or document content, said first data item comprising~~ a plurality of segments, each of said segments consisting of a corresponding sequence of bits, obtaining, ~~determining a segment identifier~~ for each of said segments, a corresponding segment identifier, each said segment identifier being based at least in part on, and being determined as a first function of the bits of ~~data comprising~~ the corresponding segment, wherein two identical segments will have the same segment identifier as determined using said first function; and
>
> ~~determining~~ ascertaining whether or not any of said segment identifiers of said plurality of segments correspond to any of a plurality of identifiers, said plurality of identifiers corresponding to a plurality of data items, each of said plurality of identifiers being based, at least in part, on said first function of the data of a corresponding segment of one of the plurality of data items ~~first data item is the same as a second data item based at least in part on said segment identifiers~~; and
>
> based at least in part on said ascertaining, determining whether or not said first data item corresponds to any of the plurality of data items.

(Amend., Dec. 30, 2010 at 10, emphasis in original; Ex. 1026.) In addition, the applicants argued that "[n]othing in Squibb teaches or in any way suggests using segment identifiers to determine whether or not said first data item corresponds to any of [a] plurality of data items." (*Id*. at 18.)

The applicants submitted a supplemental amendment[8] in which they added new claims, including claim 26 (re-numbered claim 1 when the patent issued). (Suppl. Amend. of Feb. 1, 2011 at 8-9; Ex. 1027.) No substantive remarks were provided.

The original claim set was subsequently canceled by examiner's amendment, and the new claims were allowed. (Notice of Allow., Apr. 4, 2011; Ex. 1028.)

V. THE CHALLENGED CLAIMS ARE UNPATENTABLE

A. There is Nothing New About Using Content-Based Identifiers to Determine Whether Two Data Items Are the Same

Claim 1 of the '544 patent is reproduced below.

[8] This was filed more than six months after the mailing date of the office action and thus appears not to have complied with MPEP 714.03(a).

1. A computer-implemented method, the method comprising:

(A) for a first data item comprising a first plurality of parts,

(a1) applying a first function to each part of said first plurality of parts to obtain a corresponding part value for each part of said first plurality of parts, wherein each part of said first plurality of parts comprises a corresponding sequence of bits, and wherein the part value for each particular part of said first plurality of parts is based, at least in part, on the corresponding bits in the particular part, and wherein two identical parts will have the same part value as determined using said first function, wherein said first function comprises a first hash function; and

(a2) obtaining a first value for the first data item, said first value obtained by applying a second function to the part values of said first plurality of parts of said first data item, said second function comprising a second hash function;

(B) for a second data item comprising a second plurality of parts,

(b1) applying said first function to each part of said second plurality of parts to obtain a corresponding part value for each part of said second plurality of parts, wherein each part of said second plurality of parts consists of a corresponding sequence of bits, and wherein the part value for each particular part of said second plurality of parts is based, at least in part, on the corresponding bits in the particular part of the second plurality of parts; and

(b2) obtaining a second value for the second data item by applying said second function to the part values of said second plurality of parts of said second data item; and

(C) ascertaining whether or not said first data item corresponds to said second data item based, at least in part, on said first value and said second value.

('544 patent, col. 38, l. 34 – col. 39, l.3; Ex. 1001.) At first glance, claim 1 may seem long and complicated. However, in reality, it is simple. It requires little more than obtaining "values" for two data items (each a "hash of hashes"), and comparing these values to ascertain whether the two data items correspond to each other (*e.g.*, whether they are the same). Using the letter notation in the claim itself as a guide, portion (A) obtains a first value (*i.e.*, what the specification calls an "identifier") for a first multi-part data item (*i.e.*, what the specification calls a "compound data item"). Portion (B) does the same for a second multi-part data item. Portion (C) ascertains whether or not the two data items correspond to each other (*e.g.*, whether or not they are the same) based on the first and second values. Each "value" (*i.e.*, identifier) is based on a function of the "part values" for the parts of the data item (*e.g.,* a "hash of hashes"). To be clear, portion (a1) forms a "part value" for each part in the first data item based on a "function" of the bits in the part (*e.g.*, a hash), and portion (a2) forms a "first value" for the data item as a whole based on a "function" of the part values (i.e., a "hash of those hashes"). The same operations happen for the second data item in portions (b1) and (b2) of the claim.

The applicants stated that they were entitled to this claim because "[i]n ***all*** of the prior data processing systems, the names or identifiers provided to identify data items . . . are ***always*** defined relative to a specific context," and "there is ***no direct***

relationship between the data names and the data item." ('544 patent, col. 2, ll. 27-32, col. 2, ll. 40-41, emphasis added; Ex. 1001.)

These representations were simply wrong. Prior data processing systems ***did use*** identifiers based on the contents of a data item or its segments – and not the context or pathname – ***including*** identifiers based on a "hash of hashes." In fact, these techniques were old and widely used. This is not surprising. The concept of using a mathematical function to create a "fingerprint" or "signature" for a data item based on the content of the data item predates the '544 patent by decades. For example, IBM developed one of the first hash tables in the 1950s (*see*, *e.g.*, G. D. Knott, "Hashing functions", The Computer Journal 18 (1975), vol. 3, at 274 (discussing "history of hashing"); Ex. 1011), and Professor Ron Rivest of MIT introduced the MD5 hash algorithm referenced in the '544 patent in the early 1990s. (*See, e.g.*, R. Rivest, "The MD5 Message-Digest Algorithm," Internet RFC 1321 (Apr. 1992); Ex. 1012.)

Moreover, Professor Ralph Merkle and others were partitioning data items into parts, calculating identifiers for the parts using a hash function, and then "hashing the hashes" to create a top-level signature (*i.e.*, identifier) for the data

item as a whole by the 1970s.[9] (*See* U.S. Patent No 4,309,569 to Merkle, "Method of Providing Digital Signatures," filed Sept. 5, 1979, at col. 2, ll. 54-68 and Fig. 1 (calculating signatures for a "vector of data items" by calculating signatures for parts of the vector using a hash function, then combining the signatures using the same hash function) ("Merkle"); Ex. 1031.) "Hash trees" – also known as "Merkle trees" – were well known in the field long before the '544 patent.

These hashing functions take as input the data contained in a file, segment, or other data item, and produce a much smaller-sized output value, commonly called a "hash," "hash value," "message digest" ("MD"), or "checksum." (*See, e.g.*, McGraw-Hill Dictionary of Scientific and Technical Terms, (4th ed., 1989), at 860; Ex. 1013; *see also* B. Kaliski, "A Survey of Encryption Standards," IEEE Micro (Dec. 1993), pp. 74–81, at 77; Ex. 1014.) For example, a file that is a million bytes (or even much larger) in size can be used as input to produce a hash value that is a mere 16 bytes in length. Because of the mathematical properties of the function, the odds that two different files will produce the same 16 byte hash

---

[9] The idea of partitioning data into smaller parts (*e.g.*, "pages" or "blocks") has been known for decades. (*See, e.g.,* B. Lampson and R. Sproull, "An Open Operation System for a Single-User Machine," ACM Operating System Review (Dec. 1979) at 100; Ex. 1032; *see also* A. Tanenbaum, "Operating Systems Design and Implementation," Prentice-Hall (1987) at 256-57; Ex. 1033.)

are extremely small: for example, with a 16 byte hash output, the odds that two randomly picked inputs have the same hash are 2^{-64}, or approximately one in sixteen billion billions. (B. Kaliski at 77; Ex. 1014.) Consequently, hashes are known as "signatures" or "fingerprints" because they identify data items with high reliability, just like signatures or fingerprints identify people with a high degree of certainty. (*See* McGregor and Mariani, "'Fingerprinting' – A Technique for File Identification and Maintenance", 12 Software Practice & Experience 1165 (1982) at 1165 ("fingerprinting" technique "produce[s] a quasi-unique identifier for a file, derived from that file's contents. . .[t]he idea is to provide an identifying feature for every file, which is intrinsically distinctive, and analogous (hopefully) to a human's fingerprint."); Ex. 1017.)[10]

---

[10] This reference was central to the rejection of EP counterpart application EP0826181A1 with claims having a central feature of content-based identifiers. (Annex to the communication, May 8, 2009; Ex. 1020.) The applicants amended the claims to emphasize a "licensing" limitation not found in the challenged claims (Reply to communication from the Examining Division, Nov. 18, 2009, at 4; Ex. 1021.), but this too was found unpersuasive and the rejection was maintained by the EPO. (Annex to the communication, March 14, 2012 at 4; Ex. 1022.) Following this rejection, Applicants withdrew the application from consideration. (Closing of Application, June 14, 2012; Ex. 1023.)

Although the applicants suggested in their patent application that they were the first to utilize hash functions to identify data items for file management applications, others working in the field used them for the same purposes more than a decade before the '544 patent. For example, at least sixteen years before the '544 parent was filed, researchers were already using hash functions to compare records to see if they were identical, and to eliminate duplicate. (*See*, *e.g.*, Babb, "Implementing a Relational Database by Means of Specialized Hardware", 4 ACM Transactions on Database Systems 1 (March 1979) at 2-4; Ex. 1034; Bitton and DeWitt, "Duplicate Record Elimination in Large Data Files", 8 ACM Transactions on Database Systems 255, 256 (1983) (commenting on the work of Babb); Ex. 1035.) File "fingerprinting" has long been known as a technique to identify files, and to check for duplicates. (*See* Rabin, "Fingerprinting by Random Polynomials", Center for Research in Computing Technology, Harvard University, Report TR-15-81 (1981) at 1 and 9; Ex. 1015; *see also* Manber, "Finding Similar Files in a Large File System", Department of Computer Science, University of Arizona, Report TR 93-33 (1993) at 3 (commenting on work of Rabin); Ex. 1016; McGregor and J.A. Mariani at 1165; Ex. 1017.)

Many other printed publications and patents disclose and use identifiers exactly like those described and claimed in the '544 patent, including "hashes of hashes," for exactly the same purposes. These publications disclose identifiers that

are location- and context-independent, that are determined using only the contents of a data item or the parts of a data item, and that are formed using identical algorithms to those mentioned in the '544 patent.

Kantor: For example, Dr. Frederick W. Kantor, a physicist from Columbia University, described a product called FWKCS that created "contents-signatures" for files based on their content. (Kantor at Preface 2; Ex. 1004.)[11] FWKCS used these contents-signatures to uniquely identify and compare files on a bulletin board system ("BBS"), an online file system considered a precursor to the World Wide Web. (*Id.* at Preface 2.) The contents-signature, as the name suggests, was based on a hash of the data contained in a file, just like the identifiers in the '544 patent. (*Id.* at 8; *compare* '544 patent, col. 13, ll. 31-42 and Figure 10A; Ex. 1001).

Kantor specifically addressed the issue of compound data items, in particular, "zipfiles" containing a set of files meant to be kept together. (Kantor at Preface 1; Ex. 1004.) FWKCS created "zipfile contents-signatures" for these zipfiles, based on a hash of the contents-signatures of the files within the zipfile (*i.e.,* a "hash of hashes"), just as in the '544 patent. (Kantor at 9; Ex. 1004;

---

[11] The three-page Preface section of Kantor's FWKCS user manual does not have individual page numbers. Citations to the Preface are labeled "Preface" to denote pages 1-3 of the Preface section. Otherwise, citations refer to the page numbers in the top-right margin of the remainder of the user manual.

compare '544 patent, col. 13, ll. 43-61 and Figure 10B; Ex. 1001.) FWKCS's contents-signatures could be used to identify and compare both compound data items, like zipfiles, and the parts (*i.e.*, individual files) contained within them. Kantor at Preface 2; Ex. 1004.) FWKCS used these contents-signatures for various purposes including, for example, to detect duplicate files. (*Id.* at Preface 2, 5.) If two zipfiles had the same zipfile contents-signature, they were almost certain to contain an identical set of individual files within them. *(Id.* at Preface 2.) FWKCS accordingly used the zipfile contents signatures to ascertain whether the data items corresponded.

Browne: A group of researchers at the University of Tennessee and Bell Laboratories developed yet another example of context- and location-independent identifiers for the same purposes as the '544 patent. These researchers disclosed a system that created "location-independent file names" (or "LIFNs") to identify and compare files on the Internet. (Browne at 3; Ex. 1002.) LIFNs – like the identifiers in the '544 patent – uniquely identified files based on their contents, not their locations. (*Id*. at 3; *compare* '544 patent, col. 33, ll. 28-30 (True Names used to identify files "independent of their present location"); Ex. 1001.) LIFN <signatures> were computed as "the ascii form of the MD5 signature of the file" – the same function identified in the '544 patent. (Browne at 6; Ex. 1002; *compare* '544 patent, col. 12, ll. 45-47 (using MD5 or SHA); Ex. 1001.)

Browne, like Kantor, specifically addressed compound data items (which she called "resources"), including multiple files meant to be used together, such as a software package of computer program files. (Browne at 2, 5; Ex. 1002; *see also* '539 patent, cl. 9 (referencing a "software package".) To handle these compound resources, each component (*e.g.*, each file) of the resource was assigned its own LIFN <signature>, computed as the MD5 hash of the contents of the component. (Browne at 5–6; Ex. 1002.) The LIFNs for the individual components were then grouped together in a sequence, and a LIFN <signature> was computed for the resource as a whole by performing an MD5 hash of the sequence of LIFNs for the individual components. (*Id*. at 6.) In other words, hashes were computed for the data in each component file, each hash acting as an identifier for its associated file, and a "hash of hashes" was then computed for the package, acting as the identifier for the package as a whole.

Browne used these LIFN identifiers for various purposes including to uniquely locate files on the network, and to verify the integrity of files using the well-known properties of MD5 signatures. To perform such an integrity check, a client would compute a LIFN <signature> for the file, and compare it to the LIFN <signature> at the server; if these signatures matched, the client would know that the file was intact. As Browne explained, "a client may perform an integrity check on a retrieved file by computing the signature for the file and comparing it with the

one known to be associated with the file's purported LIFN." (Browne at 5; Ex. 1002.) Browne accordingly used the LIFNs to ascertain whether or not the files corresponded.

Langer: Another researcher, Albert Langer, also addressed the same problems as the '544 patent and, like Kantor and Browne, proposed the same solution. (Langer; Ex. 1003.) Langer was particularly concerned with sharing content on the Internet prior to the rise of the World Wide Web, through the use of popular protocols such as the File Transfer Protocol (FTP). FTP sites, among other things, could be accessed to provide a listing of available files at the site, and a user could select and download files from the site. (*See, e.g.*, P. Deutsch et al., "How to Use Anonymous FTP," Internet RFC 1635 (May 1994); Ex. 1041.) Langer specifically addressed the problem of "uniquely identifying files which may have different names and/or be in different directories on different systems," and like the '544 applicants, observed that traditional location-based identifiers do not work well for distributed systems. (Langer at 3; Ex. 1003; *compare* '544 patent, col. 2, ll. 45-54; Ex. 1001.) Langer's solution, like the '544 patent, was to "provide a unique identifier for each file which is independent of location." (Langer at 3; Ex. 1003; *compare* '544 patent, col. 3, ll.53-59; Ex. 1001.) Specifically, Langer disclosed "defining a unique identifier that does NOT include a particular site identifier," by "using a cryptographic hash function such as MD5,"

once again, the same algorithm used in the '544 patent. (Langer at 4; Ex. 1003; *compare* '544 patent, col. 12, ll. 45-47; Ex. 1001.) Langer, like Kantor and Browne, also addressed the issue of compound data items, including, for example, archived files that were part of the same package. (Langer at 5; Ex. 1003.) Langer extracted each file from the archive, and computed an identifier for it based on an MD5 hash of the contents of the file. *(Id.)* He then concatenated those identifiers together to create a new file (*i.e.*, a file of the sequence of MD5 hashes), and computed an MD5 hash of the contents of the new file (*i.e.*, a "hash of hashes") to serve as an identifier for the package as a whole. (*Id.*)

Langer used these identifiers for various purposes including, like Browne, to verify the integrity of a file. As he explained: "for any users that DO wish to check validity, [the MD5 hash function] provides a VERY secure means of ensuring they have got an uncorrupted version of the specific file they were told about, regardless of where they can get it from." (*Id*. at 4.)

Woodhill: The Woodhill patent provides still another example of the use of context- and location-independent identifiers for the same purposes as the '544 patent. Woodhill created a distributed storage system that used "Binary Object Identifiers" to identify and compare files, among other functions. (Woodhill at col. 2, ll. 10-38; Ex. 1005) As Woodhill explains, these Binary Object Identifiers provided a "unique identifier for each binary object to be backed up." (*Id*. at col. 4,

lines 45-47.) The Binary Object Identifiers included three fields – a CRC value, a LRC value, and a hash value – "[e]ach . . . calculated from the contents of each binary object." (*Id*. at col. 7, l. 64 – col. 8, l. 4.)

For larger files, the binary objects could be further "granularized," that is, divided into smaller segments called "granules." (*Id*. at col. 15, ll. 9-24.) Woodhill computed a "contents identifier" for each of the granules based on a CRC value and a hash value "calculated against the contents of the 'granule.'" (*Id*. at col. 15, ll. 24-28.) He then created a "shadow file" containing the contents identifiers for each granule in a binary object, which like all files was converted into a binary object for back up. (Woodhill at col. 15, ll. 20-24; *see also* col. 2, ll 13-29, col. 5, ll. 61-63, col. 18, ll. 11-19; Ex. 1005.) Successive versions of a shadow file (like any file) could be compared using Binary Object Identifiers (i.e., a "hash of hashes," in the case of shadow file binary objects). (*Id*. at col. 2, ll. 24-38.)

These prior art references provide just a handful of many examples of the use of content-based identifiers, including "hashes of hashes," to perform basic file management functions. Indeed, the application of hash-based identifiers to these functions was so obvious that at least one commentator not only described the applications as "easy" but also posted these ideas publicly "to impede anyone who might independently have had the idea from patenting it." (Williams, "An

algorithm for matching text (possibly original)", posted to the "comp.compression" newsgroup on January 27, 1992; Ex. 1037; *see also* R. Williams, "An Introduction to Digest Algorithms," Rocksoft (Nov. 1994), at 13 (further describing potential uses for file management purposes of identifiers based on the hash of the contents of a block of data); Ex. 1038.)

In short, other than perhaps coining a new phrase – i.e., True Name – for a very old concept, there is absolutely nothing new disclosed or claimed in the '544 patent concerning the use of location-independent, content-based data identifiers.

VI. SPECIFIC GROUNDS FOR PETITION

Pursuant to Rule 42.104(b)(4)-(5) and Practice Guide Fed. Register Vol. 77, No. 27, page 6873 Petitioners have submitted claim charts in connection with this Petition (attached as Exhibits 1029, 1030, 1036, 1040), from the pending litigation between Petitioner and PersonalWeb Technologies LLC. Those charts set forth Petitioners' position with respect to those references and demonstrate that the challenged claims are anticipated and/or unpatentable in view of each of them. Petitioner also submits herewith the Declaration of Dr. Douglas Clark (Ex. 1009), a Professor of Computer Science at Princeton University. Dr. Clark confirms that the charts identify representative subject matter in each reference that teaches each and every limitation of the challenged claims. He likewise confirms how each claim is anticipated or, at a minimum, rendered obvious by the prior art.

A. Grounds of Invalidity for Challenged Claim 1 based on Kantor as a Primary Reference

Ground 1: Kantor Anticipates Challenged Claim 1

Kantor was not cited to the USPTO and not considered during prosecution of the '544 patent. It is prior art under at least 35 U.S.C. § 102(b) and anticipates claim 1 of the '544 patent.

Kantor is a published manual that describes a software program called the Frederick W. Kantor Contents-Signature System Version 1.22 ("FWKCS"). (Kantor at Title Page; Ex. 1004.) Like the '544 patent, Kantor explicitly recognizes the shortcomings of context- or location-dependent file identifiers. (Kantor at Preface 1; Ex. 1004; *compare* '544 patent, col. 3, ll. 31-45; Ex. 1001.) These include the "problem of duplicate files on electronic bulletin board systems" or BBSs.[12] (Kantor at Preface 1; Ex. 1004.) BBS users would unwittingly or intentionally upload files to a bulletin board, which the bulletin board already had. (*Id.*) Consequently, bulletin board operators "were paying for hardware to provide the capacity for these spurious [duplicate] files, and spending many hours trying to find and delete them." (*Id.*)

[12] Before the World Wide Web, computers "dialed into" a file server or network of servers where users could exchange files or other information by uploading or downloading files.

Kantor uses the same solution that would be later proposed in the '544 patent: associating a unique identifier with the ***contents*** of a file, rather than with the ***location*** of the file. Kantor calls these identifiers "contents-signatures," and uses them to identify a file based only on the contents of a file, and not its name, location, or other characteristics. (*Id.*; *compare* '544 patent, col. 3, ll. 31-45; Ex. 1001.) These signatures can be used for various purposes, including, for example, ascertaining whether a file was already stored on the BBS system. (*See, e.g.*, Kantor at 2-3; Ex. 1004.)

FWKCS computes these contents-signature based on a function of the data in a file. (*See id.* at 7-8.) Specifically, the contents-signature is constructed with "the 32-bit CRC [cyclic redundancy check][13] of the file contents and the uncompressed file-length." (*Id.*) The CRC and the file length (*i.e.,* file size) are both a function of the data contained in the file, and two files with the same content necessarily have the same contents-signature. (*Id.*) In fact, Kantor uses the same technique as in the '544 patent, creating a contents-signature with a hash and a length value. (*Id.* at 7-8; Ex. 1004; *compare* '544 patent, col. 13, ll. 31-42 and

[13] As Dr. Clark confirms, a CRC is a well-known hash function that calculates a value as a function of the file's contents. (Clark Decl., ¶ 3; Ex. 1009.)

Figure 10A; Ex. 1001.) Each contents-signature is location-independent and independent of the file's pathname, location, or context.

Kantor also creates identifiers for compound data items in the same manner as the '544 patent. Kantor explains that BBS users often bundled files into "zipfile" format, a well-known format for organizing related files into a single compound file. (*See* Kantor at Preface 2; Ex. 1004.) FWKCS generates "zipfile contents-signatures" for these zipfiles by computing the contents-signatures for each of the individual files within the zipfile, then hashing the contents-signatures for these files using an "addition modulo 2^32" hash,[14] to create the "zipfile contents-signature" for the zipfile as a whole (*i.e.,* a "hash of hashes"). (*See id*. at 9.)

---

[14] As Dr. Clark confirms, "addition modulo 2^32" is another well-known hash function that uses addition to calculate a value based on a file's contents. (Clark Decl., ¶ 4; Ex. 1009 *see also* G.D. Knott, Hashing functions, The Computer Journal 18 (1975), vol. 3, at 268 (describing "common elementary hashing functions" including addition functions); Ex.1011.) By adding together the values of the contents identifiers for the individual files, Kantor ensures that "the resulting [zipfile contents-identifier] does not depend on the names of the files, the dates of the files, [or] the order in which they appear in the zipfile . . ." (Kantor at 5, 8-9; Ex. 1004.)

FWKCS provides many operations for working with these zipfile contents-signatures. For example, FWKCS computes the zipfile and file contents-signatures for all of the zipfiles and individual files in the system, and stores them in a master contents-signature list, such as "CSLIST.SRT," which is similar to the "True File Registry" of the '544 patent. (Kantor at 18; Ex. 1004; *compare* '544 patent, col. 33, ll. 36-38; Ex. 1001). FWCKS uses the zipfile contents-signatures to delete duplicates uploaded under different names, and to determine whether a zipfile being uploaded to the system already exists in the system. (Kantor at 9; *see also* Preface 2; Ex. 1004.) As Kantor emphasizes:

> All the file_contents directly zipped in an uploaded zipfile are checked against all such file_contents in all the publicly accessible zipfiles on the system -- over two hundred thousand files, contained in over forty thousand zipfiles. If the new zipfile's contents all appear to match one_to_one the contents of a single zipfile already on the system, the zipfile is treated as a duplicate, and set aside. If each of the contained files appears to be a duplicate of a file appearing within the prior zipfiles, but they do not make a one_to_one matching with all the ones from a single zipfile, then the new file is treated as redundant, and set aside. These pieces could come from 16000 different zipfiles and still be recognized as forming a redundant composite.

(Kantor at 5; Ex. 1004.) In these examples, FWCKS compares the zipfile contents-signatures to determine if a duplicate zipfile is already present on the system. (*Id*. at Preface 2; *see also* 18, 36, 42, 45, 51, 194-95.)

FWCKS also provides a "Lookup" operation, which allows a user to avoid uploading duplicate material by checking to see whether a zipfile already exists on the system. (*Id.* at 97 and 173.) In response to such a request, FWKCS compares the zipfile contents-signature for the zipfile of interest with the zipfile contents-signatures for the zipfiles already stored on the system. (*Id.* at 96.) If a match is found, FWKCS reports back to the user that the zipfile is already present on the system. (*Id.*) Another operation, "Precheck," allows a user who receives a "new uploads" directory from a BBS "to compare the contents-signatures of files listed in it with the [contents-signatures] for all the non-directory files on his/her system, independent of filenames and comments, before deciding to download any of those newly listed files." (Kantor at 87; Ex. 1004.) This operation also can be used with zipfile contents-signatures. (*Id.* at 57-58; *see also* 33.)

As set forth in detail in the attached claim chart (Ex. 1029), and as confirmed by Dr. Clark (Clark Decl., ¶¶ 17-25; Ex. 1009), Kantor anticipates claim 1 of the '544 patent. For example, claim 1 recites:

10. A computer-implemented method, the method comprising:

(A) for a first data item comprising a first plurality of parts,

(a1) applying a first function to each part of said first plurality of parts to obtain a corresponding part value for each part of said first plurality of parts, wherein each part of said first plurality of parts comprises a corresponding sequence of bits, and wherein the part value for each particular part of said first plurality of parts is based, at least in part, on the corresponding bits in the particular part, and wherein two identical parts will have the same part value as determined using said first function, wherein said first function comprises a first hash function; and;

(a2) obtaining a first value for the first data item, said first value obtained by applying a second function to the part values of said first plurality of parts of said first data item, said second function comprising a second hash function;

(B) for a second data item comprising a second plurality of parts,

(b1) applying said first function to each part of said second plurality of parts to obtain a corresponding part value for each part of said second plurality of parts, wherein each part of said second plurality of parts consists of a corresponding sequence of bits, and wherein the part value for each particular part of said second plurality of parts is based, at least in part, on the corresponding bits in the particular part of the second plurality of parts; and

(b2) obtaining a second value for the second data item by applying said second function to the part values of said second plurality of parts of said second data item; and

(C) ascertaining whether or not said first data item corresponds to said second data item based, at least in part, on said first value and said second value.

('544 patent, col. 38, l. 34 to col. 39, l. 3; Ex. 1001.)

With regard to step (A), Dr. Clark confirms that Kantor discloses a "data item" (a zipfile) comprising a "first plurality of parts" (the files contained within the zipfile). (Clark Decl. at ¶ 19; Ex. 1009; Kantor at 2-3, 48-49; Ex. 1004.) With regard to step (a1), Dr. Clark confirms that Kantor applies a 'first function" comprising a "first hash function" (a CRC hash) to each part of the first plurality of parts (the files within the zipfile) to obtain a "corresponding part value" for each part (the contents-signatures for each file). (Clark Decl., ¶ 19; Ex. 1009; Kantor at 48-49; Ex. 1004.) Dr. Clark further confirms that each of the files in the zipfile "comprises a corresponding sequence of bits," and that each file's contents-signature is "based, at least in part, on the corresponding bits in the particular part" (it is calculated by applying a 32-bit CRC hash to the bits in the file and combining this with the file length), and that two identical files "will have the same part value

as determined using said first function." (Clark Decl., ¶ 19; Ex. 1009; Kantor at 2-3, 6-8; Ex. 1004.)

With regard to step (a2), Dr. Clark confirms that FWKCS obtains a "first value" (a zipfile contents signature) for the first data item (the zipfile), and that this value is obtained by applying a "second function" comprising a "second hash function" (an addition modulo 2^32 hash) to the part values of the first plurality of parts of the first data item (the contents-signatures of the individual files in the zipfile). (Clark Decl., ¶ 20; Ex. 1009; Kantor at 9, 55; *see also* Kantor at Preface 2, ; Ex. 1004.)

With regard to steps (B), (b1), and (b2), Dr. Clark confirms Kantor met these limitations for the same reasons as steps (A), (a1), and (a2), because Kantor computes zipfile contents-signatures for each of the zipfiles on a BBS. (*See* Clark Decl., ¶¶ 19-20; Ex. 1009; Kantor at Preface 2, 2-3, 6-9, 48-49, 55; Ex.1004.) With regard to step (C), Dr. Clark confirms that Kantor "ascertain[s] whether or not said first data item corresponds to said second data item" because, for example, FWKCS compares zipfile contents-signatures to detect and delete duplicate zipfiles uploaded under different names, to determine whether a zipfile being uploaded is already present on the system, and to process "Lookup" and "Precheck" commands. (Clark Decl., ¶¶ 22-25; Ex. 1009; Kantor at Preface 2, 5, 8, 18, 57-58, 96, 173, 194-95; Ex. 1004.) Dr. Clark further confirms that this

comparison is "based, at least in part, on said first value and said second value" (the zipfile contents-signatures of two zipfiles). (Clark Decl., ¶¶ 22-25; Ex. 1009; Kantor at Preface 2, 5, 8, 18, 57-58, 96, 173, 194-95; Ex. 1004.)

Ground 2: Kantor in View of Woodhill Renders Claim 1 Obvious

The '544 patent is clear that a "data item" can include a sequence of bits of any size including, for example, a simple or compound file, a directory file, a portion of a file, or a software product. (*See* '544 patent, col. 2, ll. 17-22, col. 5, ll. 47-51; Ex. 1001.) Nevertheless, in the event PersonalWeb contends that Kantor does not satisfy the claim limitation of a "plurality of parts" of a data item, a person of ordinary skill would have found it obvious to modify Kantor to meet that limitation. As Dr. Clark confirms, dividing a file into parts was a well-known technique to handle large files, such as databases. For example, Woodhill discloses dividing a data item into a plurality of parts (e.g., dividing files into "binary objects," and further dividing the binary object into "granules"). (Woodhill at col. 4., ll 14-30, col. 14, l.52-col. 15, l.4; Ex. 1005.) As Woodhill confirms, dividing files into smaller parts (e.g., "binary objects," and "granules") is a known and effective technique to reduce the amount of data that must be transmitted (i.e., smaller segments instead of entire files are transmitted). (Clark Decl., ¶ 26; Ex. 1009; Woodhill at col. 15, ll 4-8; Ex. 1005.)

B. Grounds of Invalidity for Challenged Claim 1 based on Browne as a Primary Reference

Ground 3: Browne Anticipates Challenged Claim 1

Browne was not referenced or discussed by the examiner during prosecution of the '544 patent.[15] It is prior art under at least 35 U.S.C. § 102(a) and anticipates claim 1 of the '544 patent.

Browne describes the Bulk File Distribution ("BFD") package developed by researchers at the University of Tennessee and Bell Laboratories as part of an effort to make scientific software easily accessible over the Internet. (Browne at 1, 6; Ex. 1002.) The BFD package is based on the concept of a "virtual repository," which is a distributed network of physical software repositories, each residing on a different file server. (*Id*. at 1–2.)

Like Kantor, Browne begins by discussing the shortcomings of context- or location-dependent file identifiers. At the time, a virtual repository could be implemented using a Uniform Resource Locator (URL) to identify each file.

[15] Browne is cited on the face of the '544 patent as one of the over 400 references. Browne played no role in the prosecution of the '544 patent.

(Browne at 2; Ex. 1002.)[16] The authors identify several problems with the use of location-based identifiers, such as URLs, to access virtual software repositories. Among other things, URLs are inadequate for ensuring the consistency of a software repository. (*Id*. at 2.) Moreover, a URL can only identify a single location; if a virtual repository offers multiple copies of the same file, each copy must be given its own URL. (*Id*. at 2.)

In order to address these shortcomings, Browne adopts the same solution that would be later proposed in the '544 patent: associating a unique identifier with the ***contents*** of a file, rather than with the ***location*** of the file. Indeed, Browne even uses the same terminology as the patent, referring to its file names as "location independent." In the BFD package, the identifier is called a "Location Independent File Name," or LIFN. (Browne at 3; Ex. 1002; *compare* '544 patent, col. 3, ll. 57-59; Ex. 1001 ("the identity of a data item is ***independent*** of its name, origin, ***location*** . . . .") (emphasis added).)

In Browne's preferred approach, the LIFN is computed as the MD5 hash of the contents of a file. (Browne at 6; Ex. 1002.) The MD5 algorithm provides a

---

[16] A URL is a character string, such as "http://www.netlib.org/index.html," that can be used to specify a transfer protocol ("HTTP"), a location ("www.netlib.org"), and a file name ("index.html"). (*See, e.g.*, T. Berners-Lee et al., "Uniform Resource Locators (URL)," Internet RFC 1738 (Dec. 1994) ; Ex. 1018.)

substantially unique fingerprint, meaning that two files with identical content will always have the same MD5 fingerprint, even if they are located on different servers, and even if the server administrators give them different names.[17] "Once a LIFN has been assigned to a particular sequence of bytes, that binding may not be changed." (Browne at 3; Ex. 1002.)

Browne also addresses compound data items (*e.g.*, related files meant to be used together) in the same manner as the '544 patent. Browne refers to these compound items as "resources," and specifically addresses the need to ensure consistency between them. (Browne at 2, 5–6; Ex. 1002.) To achieve that goal, Browne computes LIFNs for each of the components (*e.g.*, files) that make up the resource, based on an MD5 hash of the component. (*Id*. at 6.) The identifiers are then combined in a sequence to obtain a new file, called a "composite-parts-list," and a LIFN is computed for the composite-parts-list by performing an MD5 hash of the sequence of LIFNs for the individual components (*i.e.,* a "hash of hashes"). (*Id*. at 6.)

---

[17] The general syntax for the LIFN is "lifn:netlib:<signature>", referencing the file access protocol ("lifn," similar to the "http" protocol identifier in a URL), the server handling the request ("netlib"), and the unique MD5 hash used to identify a file[17] ("<signature>"). (Browne at 4, 6; Ex. 1002.)

The LIFN <signatures> are used to identify and access files, to compare them, and to verify their integrity. For example, in order to make a new file available to clients, a file server computes the file's MD5 signature and registers it with a "LIFN database." (*Id*. at 4, 6.) The registration involves sending an update to a "LIFN server" to associate the new file's MD5 signature to its physical location on the file server. (*Id*. at 4-5.) The LIFN database is a simple database of MD5 signatures and their corresponding file locations. (*Id*. at 6.) To access a file, a client asks the LIFN database to identify which servers store a file with a given MD5 signature; in response, the LIFN server provides a list of file servers. (Browne at 4–5; Ex. 1002.) The client can then download the file from one of the servers using a conventional file transfer. (*Id*. at 4.) To verify that the file actually corresponds to the file that was originally registered, the client can re-compute the MD5 signature and compare it with the one that was used to access the file; if the two signatures match, the file is intact. (*Id*. at 3, 5.)

As set forth in detail in the claim chart (Exhibit 1030), and as confirmed by Dr. Clark (Clark Decl., ¶¶ 27-33; Ex. 1009), Browne anticipates claim 1 of the '544 patent. With respect to step (A), previously quoted above in connection with the Kantor prior art, Dr. Clark confirms that Browne discloses a "data item" (a resource) comprising a "first plurality of parts" (the files contained within the resource). (Clark Decl., ¶ 29; Ex. 1009; Browne at 6; Ex. 1002.)

With respect to element (a1), Dr. Clark confirms that Browne discloses applying a "first function" comprising a "first hash function" (an MD5 hash) to each part of a plurality of parts (the files within the resource) to obtain a corresponding part value for each part (the LIFN <signatures> of the files). (Clark Decl., ¶ 29; Ex. 1009; Browne at 4; Ex. 1002.) This happens, for example, every time a file server registers a new software package with the LIFN database. (Clark Decl., ¶ 29; Ex. 1009; Browne at 4-5; Ex. 1002.) Dr. Clark further confirms that each component file or part "comprises a corresponding sequence of bits," and that "the part value for each particular part of said first plurality of parts is based, at least in part, on the corresponding bits in the particular part," because each MD5 signature is a function of the bits in the corresponding file. (Clark Decl., ¶ 29; Ex. 1009; Browne at 4; Ex. 1002.) As a result of the well-known properties of the MD5 hash function, two identical component files "will have the same [MD5 signature] as determined using said first function." (Clark Decl., ¶ 29; Ex. 1009; Browne at 4; Ex. 1002.)

With respect to element (a2), Dr. Clark confirms that Browne obtains a "first value" for the for the first data item (the LIFN <signature>) by applying a "second function" comprising a "second hash function" (an MD5 hash) to the part values of the first plurality of parts of the first data item (the LIFN <signatures> of the individual files within the resource). (Clark Decl., ¶ 30; Ex. 1009; Browne at 6;

Ex.1002.) The file server computes an MD5 signature for the resource as a whole by applying an MD5 hash function to the ordered list of MD5 signatures for the component files, previously calculated at step (a1). (Clark Decl., ¶ 30; Ex. 1009; Browne at 6; Ex. 1002.)

With respect to step (B), as Dr. Clark confirms that Browne meets these limitations for the same reasons as steps (A), (a1), and (a)(2). This step involves applying essentially the same procedure described at step (A) to a "second data item." This occurs, for example, when a file server registers a second software package with the LIFN database. (Clark Decl., ¶ 32; Ex. 1009; Browne at 4-5; Ex. 1002.) It also occurs when a client verifies the integrity of a downloaded software package by re-computing the MD5 signature from the downloaded copy. (Clark Decl., ¶ 33; Ex. 1009; Browne at 5; Ex.1002.)

With regard to step (C), Dr. Clark confirms that Browne "ascertains whether or not said first data item corresponds to said second data item in either of the two scenarios described above. (Clark Decl., ¶¶ 32-33; Ex. 1009; Browne at 5-6; Ex. 1002.) When the file server registers the second software package, the LIFN server adds the respective MD5 signature/location association to the database. (Clark Decl., ¶ 32; Ex. 1009; Browne at 6; Ex. 1002.) If the MD5 signature is already present in the database, the LIFN server just adds the new location to the existing list of locations; the server "ascertain[s] whether or not said first

[resource] corresponds to said second [resource] based, at least in part, on said first [MD5 signature] and said second [MD5 signature]." (Clark Decl., ¶ 32; Ex. 1009; Browne at 6; Ex. 1002.) Likewise, the client can verify the integrity of the downloaded package by comparing its MD5 signature to the signature previously computed by the file server, i.e., by "ascertaining whether or not said first [resource] corresponds to said second [resource] based, at least in part, on said first [MD5 signature] and said second [MD5 signature]." (Clark Decl., ¶ 33; Ex. 1009; Browne at 5; Ex. 1002.)

Ground 4: Browne in View of Woodhill Renders Claim 1 Obvious

In the event PersonalWeb contends that Browne does not satisfy the claim limitation of a "plurality of parts" of a data item, a person of ordinary skill would have found it obvious to modify Browne to meet that limitation. As Dr. Clark confirms, dividing a file into parts was a well-known technique to handle large files, such as databases. For example, Woodhill discloses dividing a data item into a plurality of parts (e.g., dividing files into "binary objects," and further dividing the binary object into "granules"). (Woodhill at col. 4., ll. 14-30, col. 14, l. 52-col. 15, l. 4; Ex. 1005.) Woodhill teaches that dividing files into smaller parts (e.g., "binary objects," and "granules") is a known and effective technique to reduce the amount of data that must be transmitted (i.e., smaller segments instead of entire

files are transmitted). (Clark Decl., ¶ 34; Ex. 1009; Woodhill at col. 15, ll. 4-8; Ex. 1005.)

C. Grounds of Invalidity for Challenged Claim 1 based on Langer as a Primary Reference

Ground 5: Langer Anticipates Challenged Claim 1

Langer was not referenced or discussed by the examiner during prosecution of the '544 patent.[18] It is prior art under at least 35 U.S.C. § 102(b) and anticipates claim 1 of the '544 patent.

Langer addresses the problem of distributing files over the Internet. Langer predates the advent of the World Wide Web, and therefore focuses on earlier file distribution technologies, notably the *File Transfer Protocol* (FTP) and the *Archie* and *WAIS* search engines. (Langer at 2; Ex. 1003.) Langer provided his contribution to the "alt.sources.d" and "comp.archives.admin" Usenet newsgroups. At the time, Usenet was one of the most effective channels for researchers to discuss current technical issues and to distribute research materials.

Like Browne, Langer recognizes the limitations inherent in the use of context- or location-based file identifiers, and the benefits of "uniquely identifying files which may have different names and/or be in different directories on different

[18] Like Browne, Langer is cited on the face of the '544 patent as one of over 400 references. Langer played no role in the prosecution of the '544 patent.

systems." (Langer at 3; Ex. 1003.) For example, identifiers that are tied to a physical server do not allow a user to select another site that is physically closer. (Langer at 3; Ex. 1003.)

Langer's solution is exactly the same as the '544 patent: determine a substantially unique identifier for each file based on the ***content*** of the file rather than its ***location***, and associate that file with the unique identifier (Langer at 3–4; Ex. 1003; *compare* '544 patent, col. 3, ll. 55-58; Ex. 1001.) Langer expressly recognizes that such an identifier may be calculated by performing a hash function on the contents of the file:

> A simple method of defining a unique identifier that does NOT include a particular site identifier would be to use a hash function on the entire contents of the file. . . . I would suggest using a cryptographic hash function such as MD5 which generates a 16 byte result.

(Langer at 4; Ex. 1003.) The'544 patent tracks Langer's solution (which predates it by almost four years) down to the choice of the same MD5 hash function.

Like Browne, Langer also specifically addresses compound data items, including, for example, archived files that are part of the same package. (Langer at 5; Ex. 1003.) Langer observes that such a package may be distributed in a variety

of archive formats, and thus files may appear to be different even though they have identical content. (Langer at 5; Ex. 1003.)

To address these compound data items, Langer divides the packages into their component files, and computes a unique identifier for each component by performing an MD5 hash on the contents of the component. (Langer at 5; Ex. 1003.) He then concatenates the identifiers for the components together in a sequence to create a new file (*i.e.*, a file of the sequence of MD5 hashes), and performs another MD5 hash on the contents of the new file (*i.e.,* a "hash of hashes") to serve as an identifier for the package as a whole. (Langer at 5; Ex. 1003.) Once again, this is the same algorithm adopted years later by the '544 patent to compute True Names for "compound data items." ('544 patent, col. 14, ll. 37-63 and Fig. 10(b); Ex. 1001.)

Langer uses these substantially unique identifiers with a central database server, such as the Archie and WAIS search engines, that associates MD5 hashes with physical locations. (Langer at 3–4; Ex. 1003.) Based on this infrastructure, a client computer can identify and compare file using their identifiers. For example, a file server computes a new file's MD5 hash and notifies a central database server of the association between that MD5 hash and the local directory path/filename of the new file. A user can query the database to find which FTP server holds a copy of a file with a specified MD5 hash. (Langer at 3–4; Ex. 1003; *see also, e.g.*,

EARN Staff, "Guide to Network Resource Tools," Internet RFC 1580 (March 1994) at 23–26 (WAIS), 29–31, 36–37 (Archie) ; Ex. 1039.) The user can then access the file from one of the previously-identified file servers, again using the file's unique identifier.[19] The same mechanism is used to access compound data items, such as archived packages. (Langer at 5; Ex. 1003.)

Langer also discloses that MD5 hashes allow clients to easily verify the integrity of the downloaded files: "for any users that DO wish to check validity, [the MD5 hash function] provides a VERY secure means of ensuring they have got an uncorrupted version of the specific file they were told about, regardless of where they can get it from. (There is currently no publicly known way to generate a file that would produce the same 16 byte MD5 code as any given file)." (Langer at 4; Ex. 1003.)

As set forth in detail in the attached claim chart (Ex. 1036), and as confirmed by Dr. Clark (Clark Decl., ¶¶ 35-41; Ex. 1009), Langer anticipates claim 1 of the '544 patent. With respect to step (A), Dr. Clark confirms that Langer

---

[19] Similarly to Browne, Langer proposes to alias MD5 signatures to actual file names on the server: "A simple ftp implementation would just hardlink every file available for ftp to a filename encoding of it's [sic] MD5 token. Users would then ftp the directory path and filename of the MD5 token and obtain the file." (Langer at 4; Ex. 1003.)

discloses a "data item" (an archive of files that are part of the same package) comprising a "first plurality of parts" (the files within an archived package). (Clark Decl., ¶ 37; Ex. 1009; Langer at 5; Ex. 1003.)

With respect to element (a1), Dr. Clark confirms that Langer discloses applying a "first function" comprising a "first hash function" to each part of a "first plurality of parts" to obtain a corresponding "part value" for each part of said first plurality of parts. Langer discloses this by computing the MD5 hash function for each of the component files, or "parts," within such a package. (Clark Decl., ¶ 37; Ex. 1009; Langer at 4-5; Ex. 1003.) Such calculation is required, for example, whenever a file server notifies the central database server of the existence of a new archive file. (Clark Decl., ¶¶ 37-41; Ex. 1009; Langer at 4-6; Ex. 1003.) For the same reason, two identical parts "will have the same [MD5 signature] as determined using said first function." (Clark Decl., ¶ 37; Ex. 1009; Langer at 4; Ex. 1003.)

With respect to element (a2), Dr. Clark confirms that Langer discloses obtaining a first value for the first data item by applying a "second function" comprising a "second function" to the part values of the plurality of parts of the first data item. (Clark Decl., ¶ 38; Ex. 1009; Langer at 5; Ex. 1003.) Langer computes the MD5 hash for the archived package as "the code obtained by applying MD5 again to the concatenation of the codes of the extracted files, in

numeric order." (Langer at 5; Ex. 1003.) Therefore, the package's identifier is computed by applying an MD5 hash function ("second function", or "second hash function") to the concatenation of the MD5 hashes for the individual files as obtained at step (a1). (Clark Decl., ¶38; Ex. 1009; Langer at 5 ; Ex. 1003.)

With respect to step (B), as Dr. Clark explains, this step is the same as step (A) for a "second data item." (Clark Decl., ¶ 37; Ex. 1009.) As with Browne, this occurs, for example, when a file server assimilates a second archive to the central database, or when a user verifies the integrity of a downloaded archive by re-computing the MD5 hash. (Clark Decl., ¶¶ 40-41; Ex. 1009; Langer at 4; Ex. 1003.)

With regard to step (C), Dr. Clark confirms that Langer meets this step in either case described above. (Clark Decl., ¶¶ 40-41; Ex. 1009; Langer at 4; Ex. 1003.) When the FTP server computes the MD5 hash of a new archive, a new entry is added to the central database server. (Clark Decl., ¶¶ 37-41; Ex. 1009; Langer at 4-5; Ex. 1003.) If the database already has an entry, the location of the FTP server is simply added to it, after "ascertaining whether or not said first [archive] corresponds to said second [archive] based, at least in part, on said first [MD5 hash] and said second [MD5 hash]." (Clark Decl., ¶¶ 40; Ex. 1009; Langer at 4; Ex. 1003.) Likewise, the user may verify the integrity of the downloaded archive by comparing its MD5 hash to the hash previously computed by the FTP

server, i.e., by "ascertaining whether or not said first [archive] corresponds to said second [archive] based, at least in part, on said first [MD5 hash] and said second [MD5 hash]." (Clark Decl., ¶ 41; Ex. 1009; Langer at 4; Ex. 1003.)

Ground 6: Langer in View of Woodhill Renders Claim 1 Obvious

In the event PersonalWeb contends that Langer does not satisfy the claim limitation of a "plurality of parts" of a data item, a person of ordinary skill would have found it obvious to modify Langer to meet that limitation. As Dr. Clark confirms, dividing a file into parts was a well-known technique to handle large files, such as databases. For example, Woodhill discloses dividing a data item into a plurality of parts (e.g., dividing files into "binary objects," and further dividing the binary object into "granules"). (Clark Decl., ¶ 42; Ex. 1009; Woodhill at col. 4., ll. 14-30, col. 14, l. 52-col. 15, l. 4; Ex. 1005.) Woodhill teaches that dividing files into smaller parts (e.g., "binary objects," and "granules") is a known and effective technique to reduce the amount of data that must be transmitted (i.e., smaller segments instead of entire files are transmitted). (Clark Decl., ¶ 42 ; Ex. 1009; Woodhill at col. 15, ll. 4-8; Ex. 1005.)

D. Grounds of Invalidity for Challenged Claim 1 based on Woodhill as a Primary Reference

Ground 7: Woodhill Anticipates Challenged Claim 1

Woodhill was cited during prosecution of the '544 patent, but there is absolutely no indication in the file history that the Patent Office considered Woodhill's granularization technique or associated shadow files, now relevant to the challenged claims.[20] Woodhill is prior art under at least 35 U.S.C. § 102(e) and anticipates claim 1 of the '544 patent.

Woodhill discloses a distributed storage management system that, as its title suggests, includes mechanisms for backing up, restoring, and accessing the files stored by each computer in the system. (Woodhill at col. 2, ll. 39-49; Ex. 1005.) Woodhill views these files "as a collection of data streams," each of which is a

[20] In the pending case of U.S. App. No. 13/352,169, family member of the '544 patent, claims similar to claim 1 of the '544 patent have been rejected using these portions of Woodhill under 35 U.S.C. § 102(e). (Non-Final Office Action, March 27, 2012, at 5; Ex. 1042.) The Patent Owner attempted to distinguish Woodhill. (Response to Non-Final Office Action, June 26, 2012, at 22; Ex. 1043.) The Patent Office responded: "Applicant argued that Woodhill does not teach 'determining a name for the second data item', 'associating the name of the second data item with the first data item as a name for the first data item'. On the contrary, Woodhill teaches at Col. 8 lines 20-65 the step of determining the hash value for the second data item and using this hash value (i.e. 'name of the second data item') as an unique identifier for the binary object (i.e. 'first data item')." (*See* Final Office Action, Aug. 1, 2012, at 15; Ex. 1044.)

"distinct collection of data within the file that may be changed independently from other distinct collections of data within the file." (*Id*. at col. 4, ll. 14-18.) The system divides each of the data streams "into one or more binary objects," having a size of one megabyte or less. (*Id*. at col. 4, ll. 12-30.) The files, and their component binary objects, are then distributed across the network. (*Id*. at col. 3, ll. 24-44.)

To identify and compare binary objects, Woodhill creates a Binary Object Identification Record, including a Binary Object Identifier, for each binary object. (*Id*. at col. 7, l. 60 – col. 8, l. 1.) The Binary Object Identifiers are "calculated from the contents of the data" in the corresponding binary object, and include a hash of the contents of the binary object. (Woodhill at col. 8, ll. 21-24; *see also* col. 7, l. 64 – col. 8, l. 31; Ex. 1005.) As Woodhill emphasizes, the "***critical feature*** to be recognized in creating a Binary Object Identifier 74 is that the ***identifier should be based on the contents of the binary object*** so that the Binary Object Identifier 74 changes when the contents of the binary object changes." (*Id*. at col. 8, ll. 58-62 (emphasis added).) In this way "duplicate binary objects, even if resident on different types of computers in a heterogeneous network, can be recognized from their identical Binary Object Identifiers 74." (*Id*. at col. 8, ll. 62-65).

Woodhill, like Kantor, Browne, and Langer, specifically addresses the issue of "compound data items." For larger files, the binary objects can be further segmented into "granules" having a size, for example, of one kilobyte. (*Id*. at col. 14, ll. 52- col. 15, l. 4.) The "granules" can then be used to track changes to binary objects at the "'granule' level." (*Id*. at col. 14, ll. 62-65.) Woodhill calculates a "contents identifier" for each of these granules based on a CRC value and hash value "calculated against the contents of the 'granule.'" (*Id.* at col. 15, ll. 21-28.) The content identifiers for each granule of a binary object are stored in a "shadow file." (*Id*. at col. col. 15, ll. 20-23.) The shadow file accordingly contains the contents identifiers for the granules in a given binary object**.**

Woodhill's "default operation is to back up all files on all disk drives 19 on the local computer 20." (*Id*. at col. 5, ll. 62-63.) Thus, shadow files, like all files stored by a local computer,[21] are "segmented into multiple binary objects" (if

[21] Woodhill, before describing in detail the operation of its Distributed Storage Manager program, notes that "the following discussion illustrates the operation of the Distributed Storage Manager program 24 on a single local computer 20." (Woodhill at col. 5, ll. 3-6; Ex. 1005.) The "Distributed Storage Manager program 24 creates a 'shadow file' which contains a 'contents identifier' for each 'granule' in the binary object currently being processed." (*Id*. at col. 15, ll. 21-24.)

necessary), and each binary object is assigned a Binary Object Identifier. (*Id*. at col. 7, ll. 40-59.)

In the particular case of backing up shadow files, Binary Object Identifiers are calculated based on the contents of shadow files, which themselves include contents identifiers for granules. (Woodhill at col. 15, ll. 20-24; Ex. 1005.) That is, the granule identifiers depend on the contents of the granule (through hash and CRC functions), and the binary object identifiers for the shadow file depends on those segment identifiers (i.e., again through hash and CRC functions).

When backing up a successive version of a file, such as a shadow file, "only those binary objects associated with the file that have changed must be backed up." (*Id*. at col. 9, ll. 6–9.) The "binary objects that have changed are identified by comparing the Binary Object Identifiers 74 calculated [during the current backup cycle] with the corresponding Binary Object Identifiers 74 associated with the next most recent Backup Instance Record 42 for the file." (*Id*. at col. 9, ll. 9-13.) Thus, Binary Object Identifiers for shadow file binary objects, which are calculated based on granule contents identifiers, are compared to determine if changes have been made to those binary objects.

As set forth in detail in the attached claim chart (Ex. 1040), and as confirmed by Dr. Clark (Clark Decl., ¶¶ 43-49; Ex. 1009), Woodhill anticipates claim 1 of the '544 patent, quoted above in Section VI.A in connection with

Kantor. With regard to step (A), Dr. Clark confirms that Woodhill discloses a "data item" (a binary object) comprising a "first plurality of parts" (the granules of the binary object). (Clark Decl., ¶ 45; Ex. 1009; Woodhill at col. 15, ll. 13-24; Ex. 1005.)

With regard to step (a1), Dr. Clark confirms that Woodhill applies a "first function" comprising a "first hash function" (a 32-bit hash) to each part of the first plurality of parts (the granules within a binary object) to obtain a corresponding "part value" for each part (the contents identifier for each granule). (Clark Decl., ¶ 45; Ex. 1009; Woodhill at col. 15, ll. 20-30; Ex. 1005.) Dr. Clark further confirms that each granule "comprises a corresponding sequence of bits" (e.g., a 1 kilobyte portion of a binary object), that each granule's contents identifier is "based, at least in part, on the corresponding bits in the particular part" (each granule's contents identifier is calculated using a 32-bit hash function on the contents of the granule), and that two identical granules "will have the same part value as determined using said first function." (Clark Decl., ¶ 45; Ex. 1009; Woodhill at col. 14, l. 52 – col. 15, l. 30; Ex. 1005.)

With regard to step (a2), Dr. Clark confirms that shadow files, like all files stored by Woodhill, are divided into binary objects to be backed up. (Clark Decl., ¶¶ 46-48; Ex. 1009; Woodhill at col. 4, ll. 13-34 and col. 5, ll. 61-63; Ex. 1005.) By calculating Binary Object Identifiers for shadow file binary objects, Woodhill

"obtain[s] a first value for the first data item" (a Binary Object Identifier for a binary object of the shadow file), and that this "first value" is obtained by applying a "second function" comprising a "second hash function" (again, a 32-bit hash function) to the part values of the plurality of parts (the Binary Object Identifier is calculated applying the hash function to the contents of a shadow file binary object, that content being granule contents identifiers). (Clark Decl. at ¶¶ 45-48; Ex. 1009; Woodhill at col. 7, l. 60 – col. 8, l. 31; *see also* col. 15, ll. 16-24; Ex. 1005.)

With regard to steps (B), (b1), and (b2), Dr. Clark confirms that Woodhill meets these limitations for the same reasons as steps (A), (a1), and (a2), because Woodhill updates shadow files and backs up these multiple successive versions (the first data item and the second data item, and their corresponding plurality of parts). (Clark Decl., ¶ 45; Ex. 1009; Woodhill at col. 9, ll. 6-28; Ex. 1005.)

With regard to step (C), Dr. Clark confirms that Woodhill "ascertain[s] whether or not said first data item corresponds to said second data item" because, by comparing binary objects of successive versions of shadow files, Woodhill by extension compares the binary objects underlying those shadow files. (Clark Decl., ¶ 49; Ex. 1009; Woodhill at col. 9, ll. 5-28; Ex. 1005.) Dr. Clark further confirms that this comparison is "based, at least in part, on said first value and said second value" (the Binary Object Identifier corresponding to a previous version of

a shadow file and the Binary Object Identifier corresponding to a current version of a Binary Object Identifier). (Clark Decl., ¶ 49; Ex. 1009; Woodhill at col. 9, ll. 5-28; Ex. 1005.)

VII. CONCLUSION

Based on the foregoing, it is clear that claims 1 of the '544 Patent recites subject matter that is anticipated by the prior art. The art cited above was not fully considered by the original Patent Examiner, and if it had been, the '544 patent would not have issued. The Petitioner requests institution of an *inter partes* review to cancel those claims.

Respectfully Submitted,

/David L. Cavanaugh/

David L. Cavanaugh
Registration No. 36,476

CERTIFICATE OF SERVICE

I hereby certify that, on December 16, 2012, I caused a true and correct copy of the foregoing materials:

- Petition for *Inter Partes Review* of U.S. Patent No. 7,945,544
- Exhibits 1001-1044
- Fee Summary Page
- EMC Corp. Power of Attorney

to be served via Federal Express on the following attorney of record as listed on PAIR:

Davidson Berquist Jackson & Gowdey, LLP

Attn: Brian Siritzky, Ph.D.

4300 Wilson Blvd., 7th Floor

Arlington, Virginia 22203

/David L. Cavanaugh/

David L. Cavanaugh

Registration No. 36,476

Table of Exhibits for U. S. Patent 7,945,544 Petition for *Inter Partes* Review

Exhibit	Description
1001.	U.S. Patent No. 7,945,544
1002.	S. Browne et al., "Location-Independent Naming for Virtual Distributed Software Repositories," University of Tennessee Technical Report CS-95-278 (Feb. 1995)
1003.	Albert Langer, "Re: dl/describe (File descriptions)," post to the "alt.sources" newsgroup on August 7, 1991
1004.	Kantor, "The Frederick W. Kantor Contents-Signature System Version 1.22," FWKCS122.REF (August 10, 1993)
1005.	Woodhill et al., U.S. Patent No. 5,649,196, entitled "System and Method For Distributed Storage Management on Networked Computer Systems Using Binary Object Identifiers"
1006.	S. Browne et al., "Location-Independent Naming for Virtual Distributed Software Repositories," http://www.netlib.org/utk/papers/lifn/main.html (Nov. 11, 1994)
1007.	K. Moore et al., "An Architecture for Bulk File Distribution," Network Working Group Internet Draft (July 27, 1994)
1008.	Chart of Patent Family Members
1009.	Declaration of Dr. Douglas Clark a Professor of Computer Science at Princeton University
1010.	Banisar et al., The Third CPSR Cryptography and Privacy Conference at 509 (1993)
1011.	G.D. Knott, Hashing functions, The Computer Journal 18 (1975), no. 3, p. 265
1012.	R. Rivest, "The MD5 Message-Digest Algorithm," Internet RFC 1321 (Apr. 1992)

1013.	McGraw-Hill Dictionary of Scientific and Technical Terms, (4th ed., 1989)
1014.	B. Kaliski, "A Survey of Encryption Standards, " IEEE Micro (Dec. 1993)
1015.	Rabin, Fingerprinting by Random Polynomials, Center for Research in Computing Technology, Harvard University, Report TR-15-81
1016.	U. Manber, "Finding Similar Files in a Large File System", University of Arizonia Technical Report (1994)
1017.	D.R. McGregor and J.A. Mariani 'Fingerprinting' – A Technique for File Identification and Maintenance, Software Practice & Experience 1165 (1982)
1018.	T. Berners-Lee et al., "Uniform Resource Locators (URL)," Internet RFC 1738 (Dec. 1994)
1019.	U. S. Patent 6,415, 280 Prosecution History, Response (August 22, 2001)
1020.	EP Pub. No. EP0826181A1 Prosecution History, Annex to the communication dated May 8, 2009
1021.	EP Pub. No. EP0826181A1 Prosecution History, Reply to communication from the Examining Division dated November 18, 2009
1022.	EP Pub. No. EP0826181A1 Prosecution History, Annex to the communication dated March 14, 2012
1023.	EP Pub. No. EP0826181A1 Prosecution History, Closing of Application dated June 14, 2012
1024.	U.S. Patent 7,945,544 Prosecution History, Application as filed on October 31, 2007
1025.	U.S. Patent 7,945,544 Prosecution History, Office Action of

	July 02, 2010
1026.	U.S. Patent 7,945,539 Prosecution History, Amendment of Dec. 30, 2010
1027.	U.S. Patent 7,945,539 Prosecution History, Supplemental Amendment of Feb. 1, 2011
1028.	U.S. Patent 7,945,539 Notice of Allowance of April. 4, 2011
1029.	Invalidity Claim Chart in view of FWKCS Contents – Signature System Version 1.22 ("Kantor")
1030.	Invalidity Claim Chart in view of LIFN ("Browne")
1031.	Merkle, U.S. Patent No 4,309,569, entitled "Method of Providing Digital Signatures," filed Sept. 5, 1979
1032.	Lampson and Sproull "An Open Operating System for a Single-User Machine," ACM (1979)
1033.	A. Tanenbaum, "Operating Systems: Design and Implementation", Prentice Hall (1987)
1034.	Babb, Implementing a Relational Database by Means of Specialized Hardware, ACM Transactions on Database Systems, Vol. 4, No.1, March 1979
1035.	D. Bitton and D. DeWitt, "Duplicate Record Elimination in Large Data Files, ACM Transactions on Database Systems, Vol. 8, No. 2, at 255 – 265 (June 1983)
1036.	Invalidity Claim Chart in view of Langer
1037.	R. Williams, "An algorithm for matching text (possibly original)", posted to the "comp.compression" newsgroup on January 27, 1992
1038.	R. Williams, "An Introduction to Digest Algorithms," Rocksoft (Nov. 1994)
1039.	EARN Staff, "Guide to Network Resource Tools," Internet RFC

	1580 (March 1994)
1040.	Invalidity Claim Chart in view of Woodhill
1041.	P. Deutsch et al., "How to Use Anonymous FTP," Internet RFC 1635 (May 1994)
1042.	Non-final Office Action dated March 27, 2012
1043.	Response to Non-Final Office Action dated June 26, 2012
1044.	Final Office action dated August 1, 2012